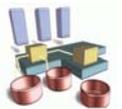


2. Klassifikation von Mehrrechner-DBS

- Merkmale für PDBS/VDBS
 - **Räumliche Verteilung**: ortsverteilt oder lokal
 - **Rechnerkopplung**: enge, lose oder nahe Kopplung
 - **Externspeicheranbindung**: partitioniert oder gemeinsam ('shared')
- Weitere Klassifikationsmerkmale
 - Funktionale Spezialisierung vs. funktionale Gleichstellung
 - Integrierte vs. heterogene/föderierte MRDBS
 - Systemansätze zur Datenintegration
- Grobbewertung von MRDBS-Alternativen
- PDBS-Architekturen
 - **Scale-Up** vs. **Scale-Out**
 - Shared-Nothing vs. Shared-Disk



Räumliche Verteilung

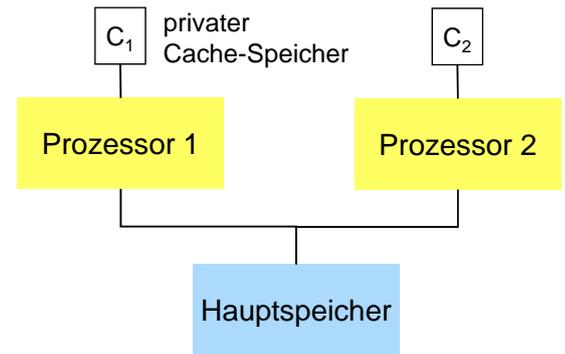
- **ortsverteilt**:
 - unterstützt dezentrale Organisationsstrukturen mit i. a. hohen Autonomieerfordernissen
 - unterstützt Katastrophen-Recovery (replizierte DB an entfernten Knoten)
 - relativ langsame Kommunikation
 - Signallaufzeiten > 100 ms
 - aufwändige Protokolle (> 10.000 Instruktionen pro Send/Receive)
- **lokal**:
 - schnelle Rechnerkopplung (gemeinsame Speicher bzw. Hochgeschwindigkeitsnetz)
 - effektive dynamische Lastverteilung möglich
 - bessere Voraussetzungen für Intra-Transaktionsparallelität
 - einfachere Administration



Enge Rechnerkopplung (tightly coupled systems)

■ Eigenschaften

- Gemeinsamer Hauptspeicher
- 1 Kopie von Software-Komponenten (BS, DBVS. Anwendungen)
- HW-Cache pro Prozessor

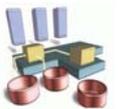


■ Vorteile:

- weit verbreitet
- wenig neue DB-Probleme (DBS-Ausführung in mehreren Prozessen)
- effiziente Kommunikation über Hauptspeicher
- Lastbalancierung durch Betriebssystem
- Single System Image

■ Nachteile:

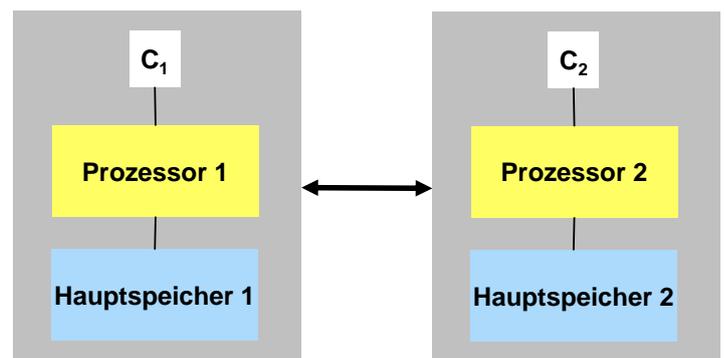
- Mangelnde Fehlerisolation
- begrenzte Erweiterbarkeit und Skalierbarkeit (meist $N < 10$)
- Cache-Kohärenz



Lose Rechnerkopplung (loosely coupled systems)

■ Eigenschaften

- N selbständige Rechner (pro Knoten eigener Hauptspeicher, eigene Software-Kopien)
- Kommunikation über Nachrichtenaustausch



■ Vorteile:

- höhere Fehlerisolation/Verfügbarkeit
- bessere Erweiterbarkeit

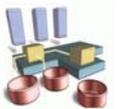
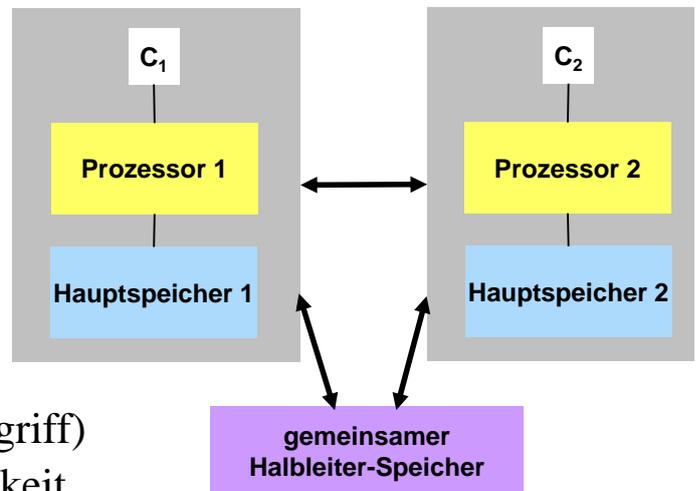
■ Nachteile:

- Nachrichtenaustausch aufwendig (Kommunikations-Overhead)
- kein 'single system image'



Nahe Rechnerkopplung (closely coupled systems)

- Kompromiß zwischen enger und loser Kopplung
 - effizientere Kommunikation als mit loser Kopplung unter Beibehaltung einer ausreichenden Fehlerisolation und Erweiterbarkeit
- Merkmale
 - N selbständige Rechnerknoten
 - gemeinsame Halbleiter-Speicherbereiche
 - lokale Rechneranordnung
- Speichereigenschaften
 - schneller, synchroner Zugriff (kein Prozeßwechsel während Zugriff)
 - i.a. keine Instruktionsadressierbarkeit
 - ggf. nichtflüchtig
- Unterstützung z.B in IBM z/OS-Mainframes



Externspeicheranbindung

gemeinsam:

jeder Prozessor kann alle Externspeicher / Daten direkt erreichen

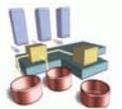
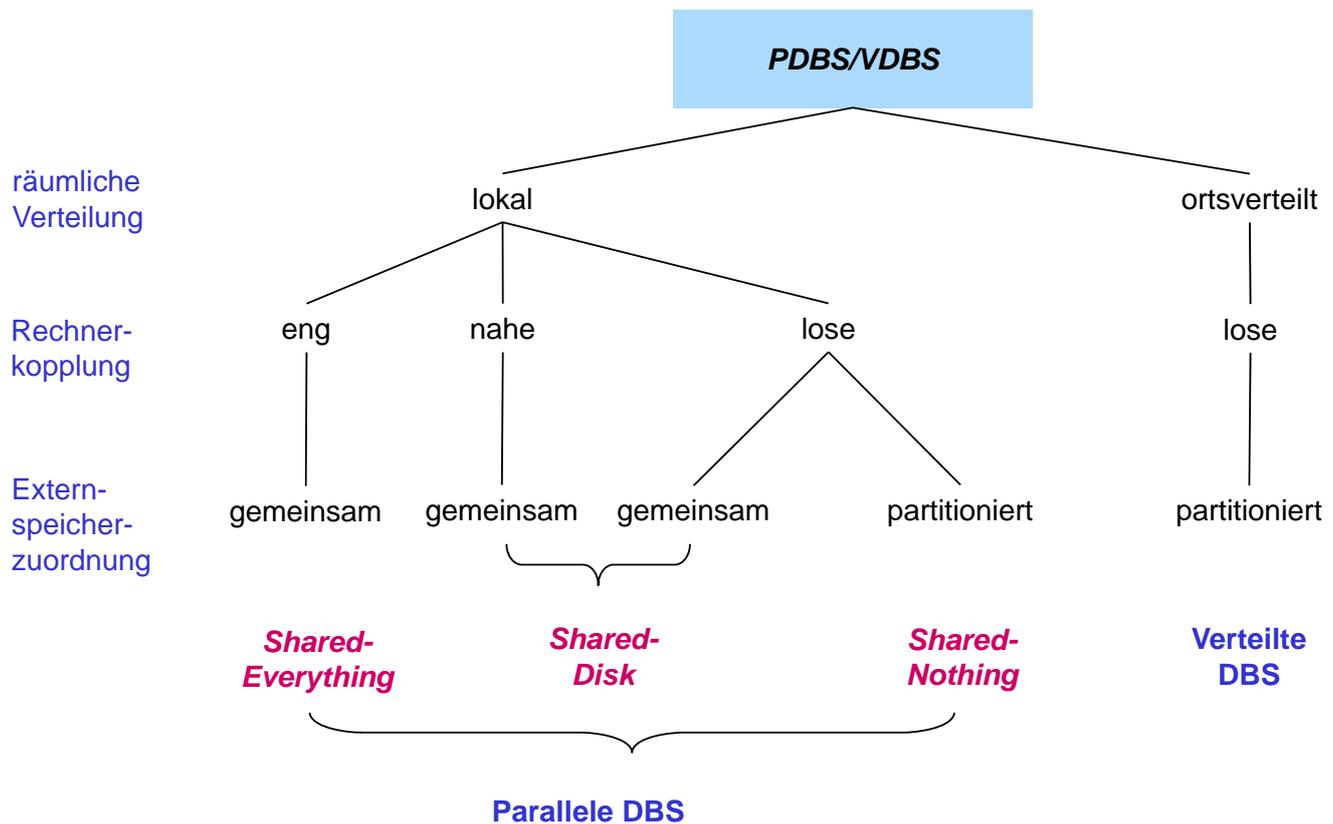
- lokale Rechneranordnung
- lose oder nahe Kopplung bzw. enge Kopplung
- hohes Potential zur Lastbalancierung

partitioniert:

Externspeicher sind primär nur je einem Knoten zugeordnet

- lokale oder ortsverteilte Rechneranordnung
- i. a. lose Rechnerkopplung
- verteilte Transaktionsausführung, um auf entfernte Daten zuzugreifen





Verteilung der Funktionalität

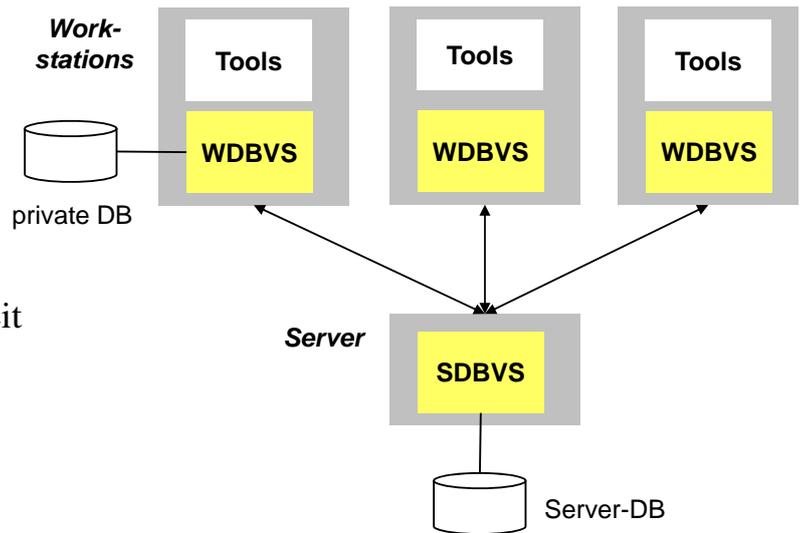
- funktionale Gleichstellung („horizontale Verteilung“)
 - jeder Knoten besitzt gleiche Funktionalität bzgl. DB-Verarbeitung
 - i.a. vollständige DBMS pro Knoten
 - Replikation der Funktionen
- funktionale Spezialisierung („vertikale Verteilung“)
 - Partitionierung von Funktionen
 - Beispiele:
 - *DB-Maschinen* mit Spezialprozessoren für bestimmte DB-Funktionen (Join-Prozessor, Sortier-Prozessor, etc.)
 - Intelligente Platten-Kontroller z.B. für Selektion/Projektion (**Oracle Exadata**)
 - Workstation/Server-DBS
 - Web-Informationssysteme (Multi-Tier-Architekturen) mit DB-Server und DB-Verarbeitung auf Applikations-Server
 - Datenintegrationsansätze, z.B. Föderierte DBS mit Query-Mediator
 - Spezialisierung erschwert Lastbalancierung, Erweiterbarkeit und Fehlertoleranz
- Mischformen: Partitionierung + Replikation von DBS-Funktionen



Workstation/Server-DBS

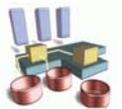
■ OODBS, Ingenieur-Anwendungen

- DB-gestützte Verarbeitung großer, komplex-strukturierter Datenmengen in der Workstation
- hohe Rereferenz-Wahrscheinlichkeit bei den Daten
- lange Transaktionen

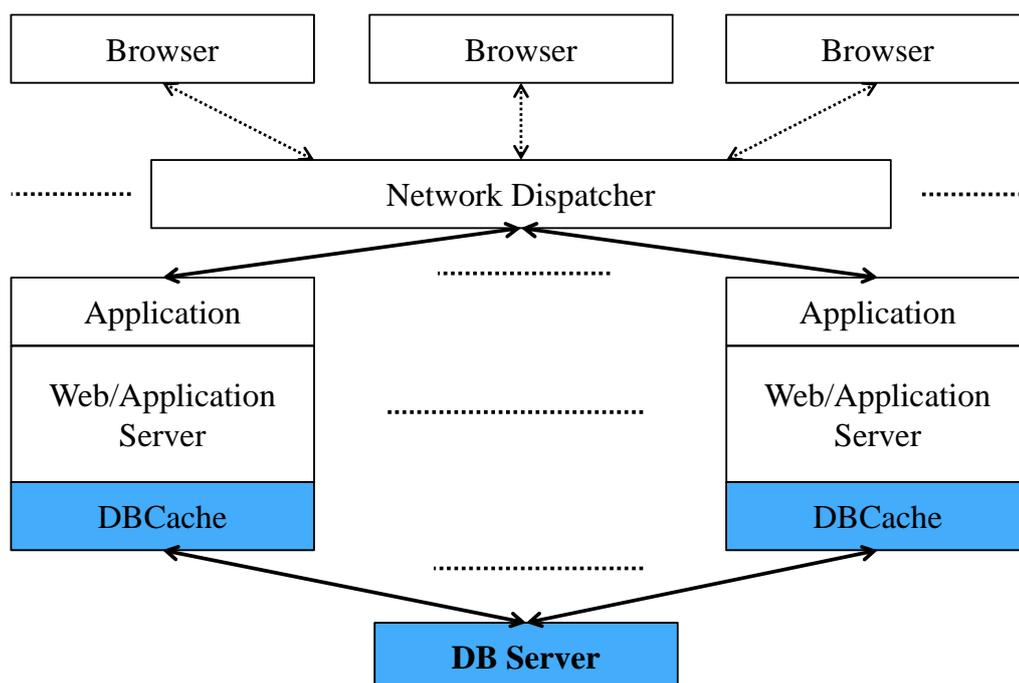


■ funktionale Spezialisierung

- Datenverwaltung in Workstation und Server
- Workstation-Objektpuffer: Einsparung von Kommunikationsvorgängen
- lokale Ausführung von Anfragen und Methoden
- globale Aufgaben auf dem Server: Logging, Synchronisation, ...



Web-Informationssysteme mit Mid-Tier Caching



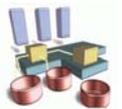
Integrierte vs. heterogene/föderierte MRDBS

■ Integrierte Mehrrechner-DBS

- 1 logische Datenbank: DB-Zugriff wie im zentralen Fall (Verteiltransparenz für AP)
- **Top-Down-Ansatz** zur Verteilung einer DB
- homogenes MRDBS (z. B. identische DBVS-Instanzen)
- geringe Autonomie für beteiligte DBVS
- Beispiele: Verteilte DBS, Parallele DBS

■ Heterogene/ föderierte Mehrrechner-DBS

- **Bottom-Up-artige Kopplung** existierender Datenbanken
- weitgehend unabhängige DBVS mit privaten konzeptionellen DB-Schemata
- partielle Zulassung externer Zugriffe (Kooperation)
- Heterogenität bei Datenmodellen und Transaktionsverwaltung möglich
- Probleme mit semantischer Heterogenität
- Verteilungstransparenz i.a. nur bedingt erreichbar
- verschiedene Alternativen zur Datenintegration



Nutzung verteilter Datenquellen

■ Suchmaschinen, v.a. für unstrukturierte Daten

- Weltweit, unternehmensweit (enterprise search), desktop search

■ Entity-Suchmaschinen

- Fokus auf bestimmte Objekttypen (Produkte, Personen, Publikationen, ...)
- Bsp.: Google Produktsuche, Google Scholar

■ Anwendungsintegration (Enterprise Application Integration)

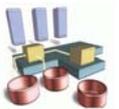
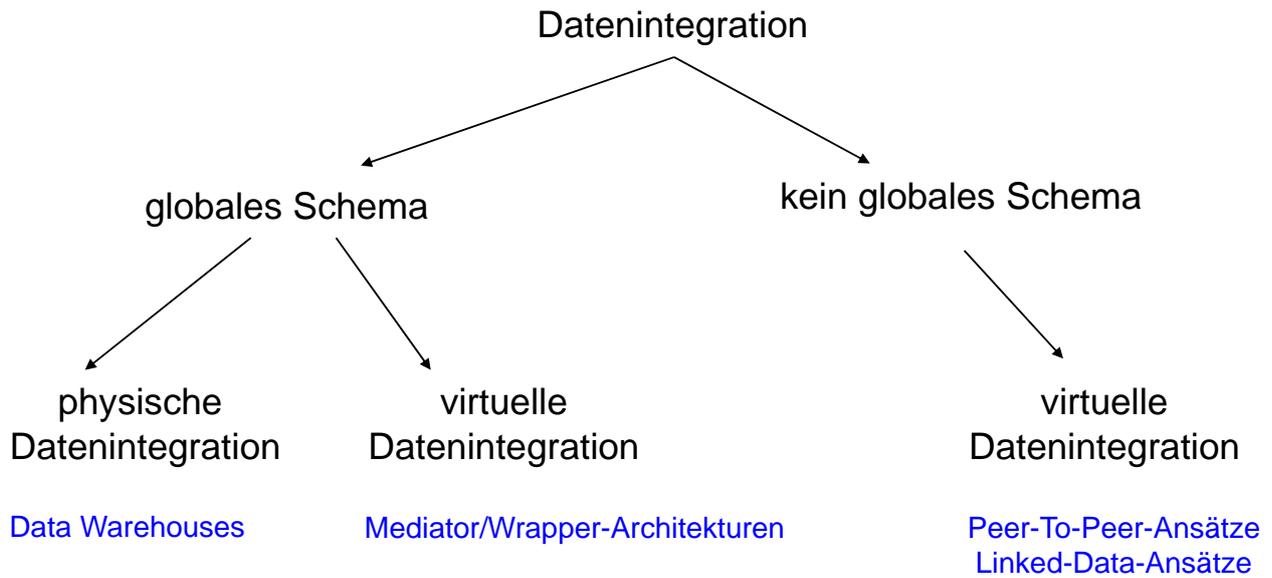
- z.B. über Web Services
- Mashups

■ Datenintegration

- Virtuelle Integration (föderierte DBS, Mediator/Wrapper-Ansätze)
- Physische Integration (Data Warehousing)
- Dezentrale Ansätze ohne globales Schema (zB Peer-to-Peer-Architekturen, Linked Data)

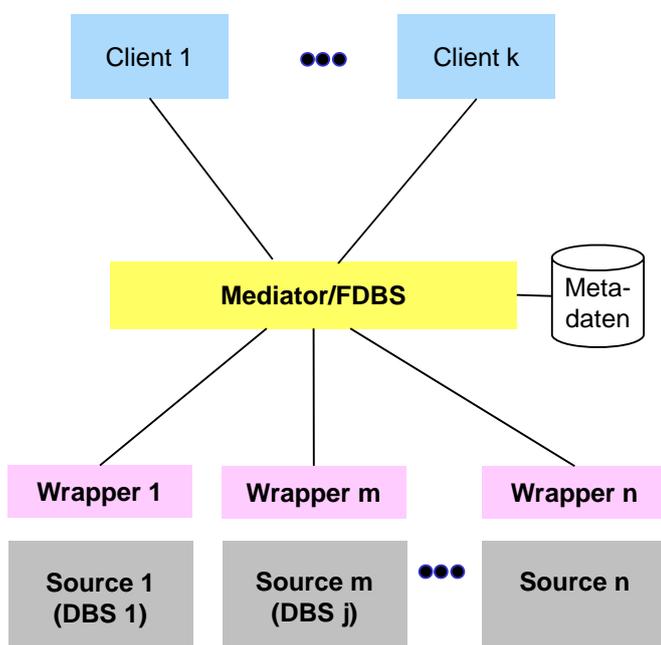


Heterogene DBS

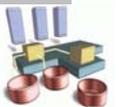
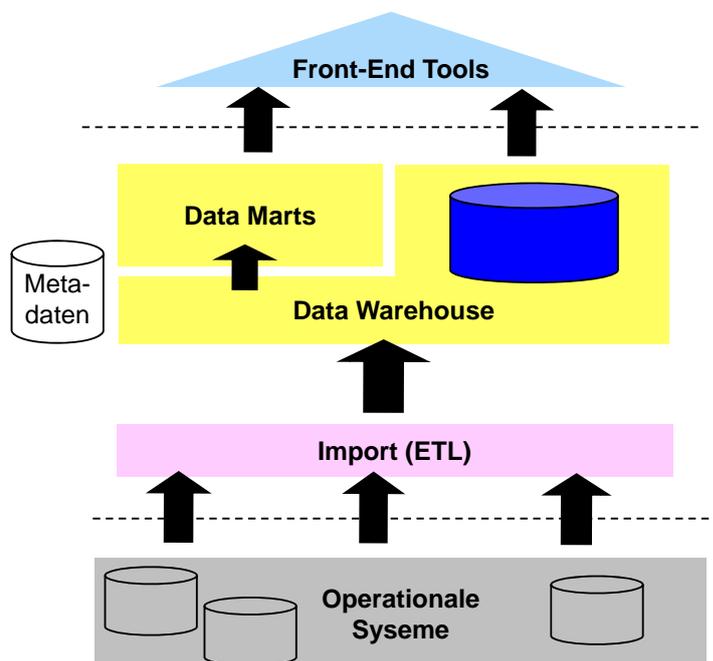


Grundlegende Alternativen zur Datenintegration

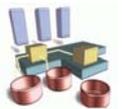
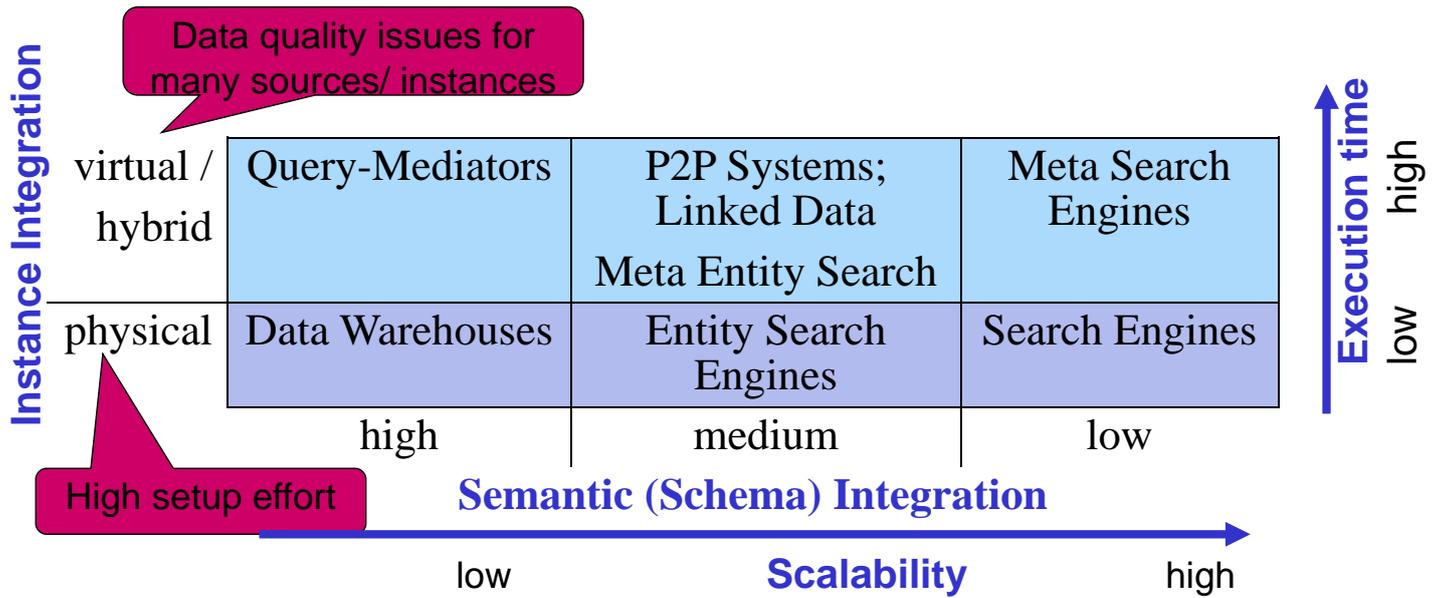
Virtuelle Integration
(Mediator/Wrapper-Architekturen, föderierte DBS)



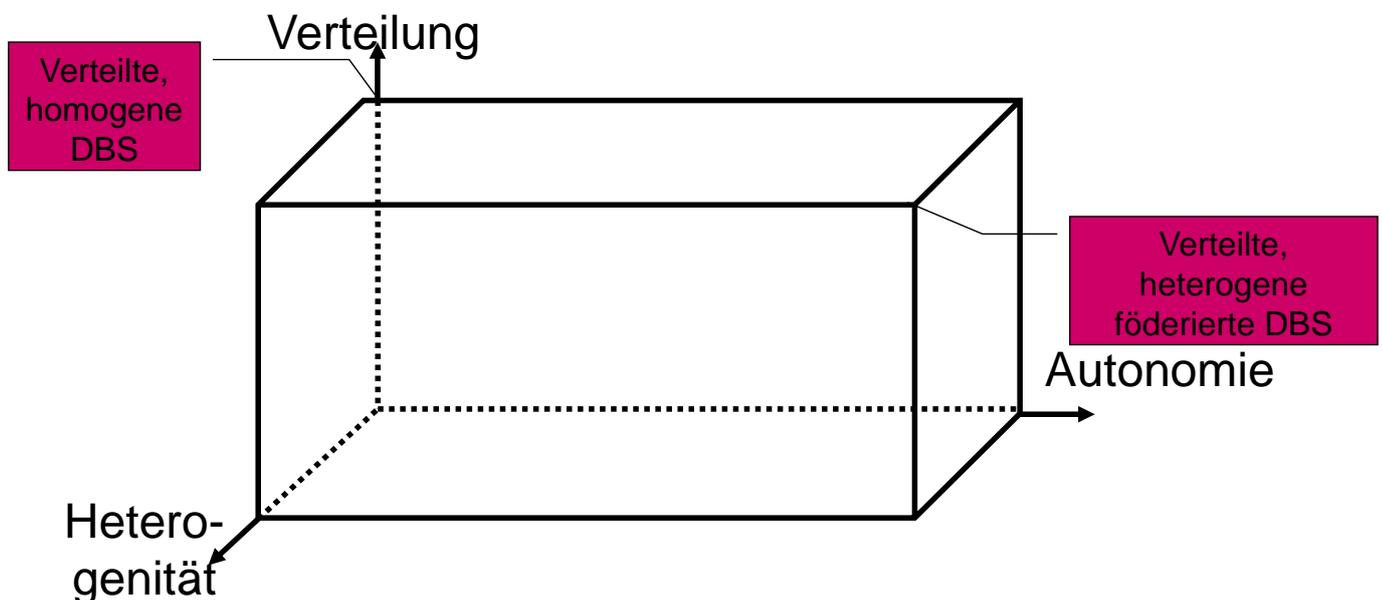
Physische (Vor-) Integration
(Data Warehousing)



Integration heterogener Daten: Alternativen



Klassifikation nach Özsu/Valduriez



Grobbewertung von Mehrrechner-DBS

	Parallele DBS (SD, SN)	Verteilte DBS	Föderierte DBS
Hohe Transaktionsraten	++	o/+	o
Intra-TA-Parallelität	++	o/+	-/o
Erweiterbarkeit	+	o/+	o
Verfügbarkeit	+	+	-
Verteilungstransparenz	++	+	o
Geographische Verteilung	-	+	+
Knotenautonomie	-	o	+
DBS-Heterogenität	-	-	+
Administration	o	-	-/--



3 Stufen der Verteilung

1. **Scale-Up:** mehrere Prozessoren innerhalb von 1 Knoten (Shared Everything)

- Sehr effiziente Kommunikation; Datenaustausch über Haupt- oder Externspeicher
- Direkter Zugriff auf gesamte Datenbank für alle DBMS-Instanzen; zentrale Datenstrukturen (Sperrtabelle, DB-Puffer, etc.)
- Vglw. einfache DB-Administration
- Wird von allen DBS-Herstellern unterstützt (Microsoft, Oracle, IBM ...)
- Begrenzte Erweiterbarkeit und Verfügbarkeit
- SMP-Leistungsfähigkeit reicht für Mehrzahl von Datenbanken
- Relativ hohe Kosten im High-End-Bereich

2. **Scale-Out:** SN/SD/Hybrid-Cluster

- Hohe Skalierbarkeit durch unabhängige Rechnerknoten (kein gemeinsamer Hauptspeicher, lokale Software)

3. **Verteiltes DB-Mirroring** (für SE, SD oder SN)



Skalierbarkeit: Scale-Up vs. Scale-Out

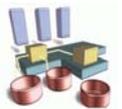


“Scale Up”

- Schnellere SMP-Knoten
- Shared Everything

“Scale Out”

- N unabhängige Rechner (z.B. Commodity-Server)
- Hinzufügen neuer Server nach Bedarf
- Shared Nothing oder Shared Disk (Cluster)



Scale-Out

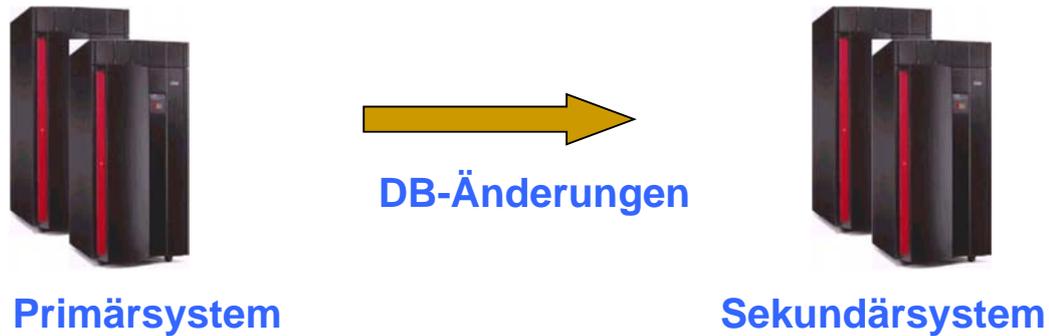
- Sehr viele preiswerte Standard-Knoten (Blades)
 - Geringer Administrationsaufwand
 - Leichte Erweiterbarkeit
 - Ausreichend für gut partitionierbare Lasten aus einfachen Operationen
 - Bsp.: Google

Vs.

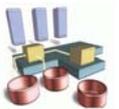
- Moderate Zahl von High-End-Servern



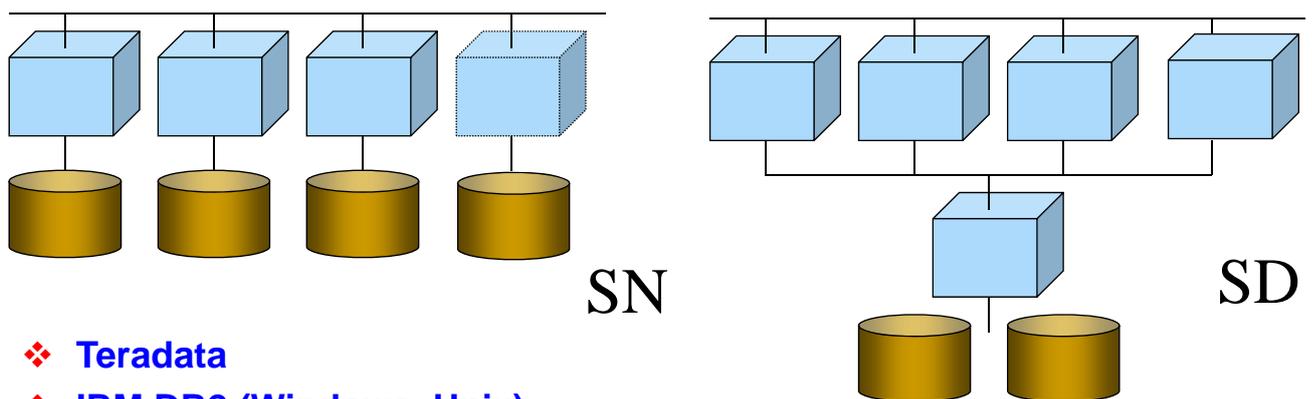
DB-Mirroring für Hochverfügbarkeit



- Komplette DB-Kopie an entferntem System
- Fortlaufende Übertragung aller Änderungen aus Primärsystem (z.B. Log-Transfer) und Anwendung auf Kopie
- Schutz auch gegenüber Katastrophen
- Anwendbar für alle PDBS-Architekturen im Primärsystem: SE (z.B. MS SQL-Server), SN, SD und Kombinationen



Shared Nothing (SN) vs. Shared Disk (SD)



- ❖ Teradata
- ❖ IBM DB2 (Windows, Unix)
- ❖ MS Parallel Data Warehouse
- ❖ viele NoSQL-Systeme

- ❖ Oracle (RAC, Exadata)
- ❖ IBM DB2 z/OS, DB2 PureScale
- ❖ SAP Sybase

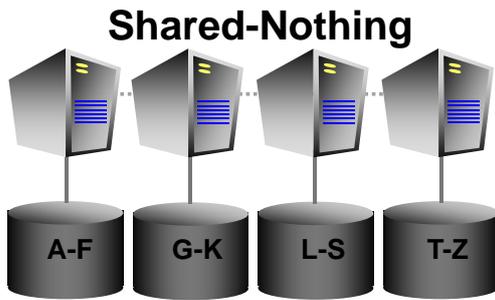
- SN auf SD-Hardware realisierbar und umgekehrt
- wesentlich ist Sicht der DBMS-Instanzen

SN: Partitionierung der Datenbank unter den DBMS-Instanzen mit daraus abgeleiteter Anfrage- und Transaktionsverarbeitung

SD: direkter Zugriff auf gesamte DB für jedes DBMS; lokale Puffer und Sperrtabellen, Datenaustausch über Externspeicher bzw. Nachrichten

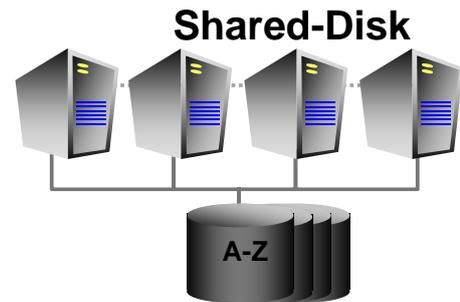


SN vs. SD: Leistungsfähigkeit



Statische Datenpartitionierung bestimmt Ausführungsort von DB-Operationen und damit Kommunikationsaufwand

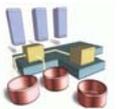
Geringe Möglichkeiten zur dynamischen Lastbalancierung



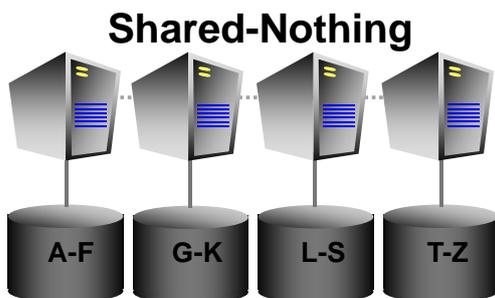
Hohe Flexibilität zur Parallelisierung und Lastbalancierung aufgrund Erreichbarkeit aller Daten von jedem Knoten

Hoher Aufwand für Synchronisation und Kohärenzkontrolle

- Lokalität ermöglicht Einsparungen
- Nahe Kopplung reduziert Overhead

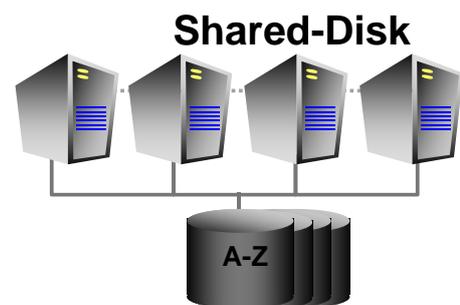


SN vs. SD: Erweiterbarkeit



Hinzufügen neuer Knoten hardwareseitig einfach (viele Knoten sind möglich)

Neuer Rechner erfordert physische Neuaufteilung der Datenbank (N -> N+1)

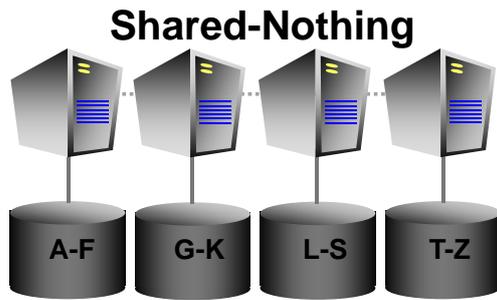


Keine physische (Neu-) Aufteilung der DB bei neuem Rechner

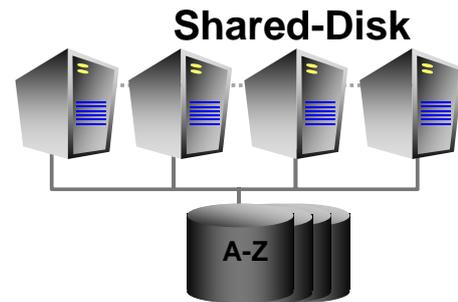
Direkte Plattenanbindung kann Rechneranzahl begrenzen



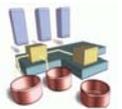
SN vs. SD: Recovery



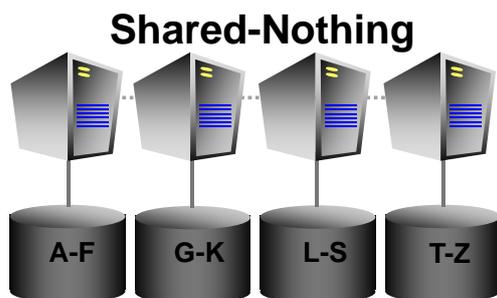
Übernahme/Recovery der betroffenen Partition durch anderen Rechner vorzusehen (ggf. Überlastungsgefahr)



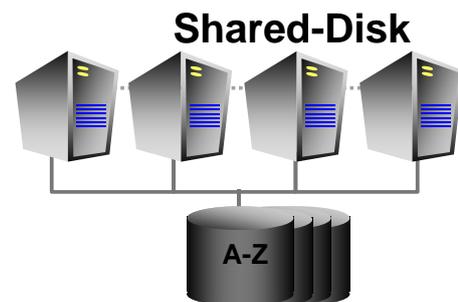
Gesamte DB bleibt nach Rechnerausfall erreichbar
Komplexe Crash-Recovery
Erstellung einer globalen Log-Datei



SN vs. SD: Technische Probleme



DB-Partitionierung bzgl. Rechner
Verteilte und parallele Anfrageverarbeitung
Verteiltes Commit-Protokoll
Globale Deadlock-Behandlung
Replikationskontrolle
Katastrophen-Recovery ...

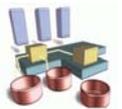


DB-Partitionierung bzgl. Platten
Parallele Anfrageverarbeitung
Globale Synchronisation
Kohärenzkontrolle
Globaler Log, Crash-Recovery
Lastbalancierung
Katastrophen-Recovery ...



Zusammenfassung

- Vielfältige Anforderungen an Mehrrechner-Datenbanksysteme führen zu verschiedenen Architekturtypen:
 - Parallele DBS, Verteilte DBS, Föderierte DBS, Data Warehouses ...
- Klassifikationsmerkmale
 - Räumliche Verteilung, Rechnerkopplung, Externspeicheranbindung, Integrierte vs. föderierte und homogene vs. heterogene DBS, funktionale Spezialisierung vs. Gleichstellung
- Parallele DBS:
 - Ziele: hohe Transaktionsraten, kurze Antwortzeiten, hohe Verfügbarkeit, Skalierbarkeit, Kosteneffektivität
 - Lokale Rechneranordnung: effiziente Kommunikation, dynamische Lastbalancierung, effiziente Parallelisierung komplexer Anfragen
 - Hauptansätze: Shared-Everything, Shared-Disk, Shared-Nothing
- Verteilte DBS: räumlich verteilte, integrierte MRDBS (globales Schema)



Zusammenfassung (2)

- Mehrrechner-DBS mit funktionaler Spezialisierung
 - z.B. Workstation/Server-DBS, Multi-Tier-Architekturen
 - Nutzung von Spezial-Hardware („Datenbank-Maschinen“) weitgehend gescheitert: geringe Kosteneffektivität, Funktionalität und Flexibilität
- Virtuelle Datenintegration
 - Föderierte DBS / Query-Mediatoren / Peer-DBS / Linked Data
 - Bewahrung einer relativ hohen Knotenautonomie
- Data Warehouses: physische Integration heterogener Datenbanken
 - Data Warehouse kann durch zentrales DBS oder PDBS verwaltet werden
- PDBS-Vergleich
 - Skalierbarkeit: Scale-Up (SE) einfacher umsetzbar als Scale-Out
 - DB-Mirroring für Hochverfügbarkeit
 - Scale-Out: Shared-Disk vs. Shared-Nothing
 - Probleme: DB-Partitionierung, Lastbalancierung, Lokalität, Administration ...

